

The Complexity of Simulation and Matrix Multiplication

Massimo Cairo*
 Università di Trento
 massimo.cairo@unitn.it

Romeo Rizzi
 Università di Verona
 romeo.rizzi@univr.it

August 31, 2016

Abstract

Computing the simulation preorder of a given Kripke structure (i.e., a directed graph with n labeled vertices) has crucial applications in model checking of temporal logic. It amounts to solving a specific two-players reachability game, called simulation game. We offer the first conditional lower bounds for this problem, and we relate its complexity (for computation, verification, and certification) to some variants of $n \times n$ matrix multiplication.

We show that any $O(n^\alpha)$ -time algorithm for simulation games, even restricting to acyclic games/structures, can be used to compute $n \times n$ boolean matrix multiplication (BMM) in $O(n^\alpha)$ time. This is the first evidence that improving the existing $O(n^3)$ -time solutions may be difficult, without resorting to fast matrix multiplication. In the acyclic case, we match this lower bound presenting the first subcubic algorithm, based on fast BMM, and running in $n^{\omega+o(1)}$ time (where $\omega < 2.376$ is the exponent of matrix multiplication).

For both acyclic and cyclic structures, we point out the existence of natural and canonical $O(n^2)$ -size certificates, that can be verified in truly subcubic time. In the acyclic case, $O(n^2)$ time is sufficient, employing standard matrix product verification. In the cyclic case, a max-semi-boolean matrix multiplication (MSBMM) is used, i.e., a matrix multiplication on the semi-ring (\max, \times) where one matrix contains only 0's and 1's. This MSBMM is computable (hence verifiable) in truly subcubic $n^{(3+\omega)/2+o(1)}$ time by reduction to (\max, \min) -multiplication.

Finally, we show a reduction from MSBMM to cyclic simulation games which implies a separation between the cyclic and the acyclic cases, unless MSBMM can be verified in $n^{\omega+o(1)}$ time.

*This work was supported by a joint PhD program with the University of Verona, Department of Computer Science, under PhD grant “Computational Mathematics and Biology”.

1 Introduction

In the context of model checking, the simulation preorder of a transition system is an abstraction that allows to reduce the state space, while preserving the satisfiability of a large class of temporal logic formulas [8]. On Kripke structures (i.e. directed vertex-labeled state-transitions graphs), it can be defined co-inductively: a state t *simulates* a state s whenever t and s are labeled in the same way and, for every transition from s to s' , there is a transition from t to t' , such that t' simulates s' .

Being a crucial problem in model checking, the computation of the simulation preorder has been studied extensively, both for explicitly defined systems and for implicit transition systems arising from process algebras. However, in the most basic setting of finite systems provided explicitly, some fundamental complexity questions about this problem are still open. In this work we address some of these questions, discovering a close relationship, in terms of complexity issues, between this problem and some variants of matrix multiplication.

Motivation. Transition systems are essentially labeled directed graphs, possibly infinite, whose vertices represent states, edges represent the possible transitions between states, and labels represent visible properties of a system, such as I/O. When reasoning about a transition system, it is possible to ignore irrelevant details by means of *abstractions*, which reduce the system to a smaller structure while preserving the properties of the system that are being studied [8]. Abstractions are commonly expressed as equivalence/preorder relations between the system states: when two states are considered equivalent according to the abstraction, they can be collapsed into one. Several abstractions have been defined in the literature [3], the most important being bisimulation equivalence [33], simulation preorder [31] and trace equivalence [25], along with their respective variants. These abstractions, listed from the finer to the coarser, preserve the validity of formulas in progressively smaller fragments of the μ -calculus logic [8].

In this work we consider finite transition systems, with n states and $m \geq n$ transitions, whose graph is given explicitly and is accompanied by labels on the states. (These vertex-labeled graphs are called Kripke structures [29].) The bisimulation equivalence, the finest among the mentioned abstractions, is computed with an almost-optimal running time $O(m \log n)$ by the algorithm of [32]. At the opposite end, the computation of the trace equivalence relation is known to be PSPACE-complete [36]. The simulation preorder lies between these two extremes: the problem is tractable, but no $o(n^3)$ -time algorithm is known (while only $\Theta(n^2)$ is needed for output).

Polynomial algorithms to compute the simulation preorder have been first presented in [6, 12, 13], improved to $O(mn)$ time independently in [24] and [7]. These algorithms perform a fixpoint computation, starting with the full relation, and then repeatedly removing pairs of states as soon as they are discovered not to be in the simulation relation. More recently, a large family of new algorithms has been proposed [22, 23, 34, 35, 30, 9] whose running time depends not only on the size of the input system, but also on the number n^* of equivalence classes in the simulation preorder relation. They work by keeping a partition of states into blocks of possibly equivalent states: working mostly at the block level, these algorithms run faster when n^* is much smaller than n . However, they still require $\Omega(n^3)$ time in the worst case, since the preorder relation could turn out to be a (non-trivial) partial order, where all the blocks eventually reduce to singletons.

A question arises naturally: is it possible to obtain a subcubic algorithm for the simulation preorder? Our work stems from the realization that this problem hides a boolean matrix multiplication inside its belly, and this explains why getting below the $\Omega(n^3)$ time barrier has been so difficult. Motivated by this result, we address the simulation preorder from viewpoint of pure computational complexity, discovering that the relationship with matrix multiplication is rich and many-sided. Moreover, we study the existence of explicit certificates for the simulation preorder, and the pos-

sibility to check the result more efficiently than computing it from scratch. Despite being crucial for a deep understanding of the algorithmic problem, to the best of our knowledge, the analysis of certificates is lacking in previous work.

Simulation as “two-tokens” games. To present our results, we first reduce the computation of the simulation preorder to its essential underlying algorithmic problem, expressed in terms of two-players *reachability games* [18, 2, 11]. While the correspondence between simulation preorder and two-players games is well-established in the literature (see, e.g., [24, 20, 10]), in our analysis we point out the specific structure of simulation games, with respect to general reachability games, and how this structure can (or cannot) be exploited algorithmically. This is very relevant for the problem: with a scrupulous eye, one can notice that the best-so-far $O(nm)$ -time algorithm for simulation does not exploit this structure at all; actually, a known linear-time algorithm [4, 2] for reachability games, when applied to simulation games, achieves the same running time (see Remark 8). To achieve any improvement along this line, the peculiarities of simulation games need to be taken into account.

In a reachability game, the goal of the first player Alice is to reach a configuration among a given set, while the second player Bob tries to avoid this. If Alice manages to reach the goal, she wins, otherwise the game continues forever and the victory is assigned to Bob. A *simulation game* is a particular reachability game, defined in terms of a given Kripke structure. A configuration consists of a pair of states (s, t) . Alice reaches her goal, and wins immediately, when s and t hold different labels. Otherwise, she first chooses a transition (s, s') from s , then Bob chooses a transition (t, t') from t , and the game moves to the next configuration (s', t') . It is well-known that t simulates s iff Alice does not have a winning strategy from (s, t) ; in fact, the latter is sometimes used as the definition of simulation preorder [24, 20].

In this paper, we define another type of reachability games, called *two-tokens reachability games* (2TRG), which generalize simulation games. In a 2TRG there are two tokens, which are moved in turn by the two players. Differently from simulation games, in 2TRGs the two tokens move along two distinct graphs. Moreover, in 2TRGs the set of goal configurations is arbitrary, and may include configurations where either player holds the turn, not necessarily Alice. Despite being more general than simulation games, we prove that 2TRGs are not computationally harder to solve, hence strictly equivalent. We analyze 2TRGs instead of simulation games, with a two-fold advantage. First, we drop some of the assumptions which are not helpful in studying the complexity of simulation games, thus reducing ourselves to a more essential algorithmic problem. Second, we introduce a symmetry between the roles of the two players which, thanks to dualization, halves the work required to describe some of our results. The reduction from 2TRGs to simulation games is quite simple, so the reader is left with the opportunity to map our results directly to simulation games with ease.

The solution of reachability games is easily characterized in terms of *closed sets* and *progress measures* (concepts similar to those used in the more involved Büchi games and parity games [26, 2]), which in turn can be defined naturally as (pre-/post-)fixpoints of some *lifting operators* (see [26, 20, 11]). When these notions are applied to 2TRGs, the relationship with matrix multiplications appears clear: the lifting operators themselves are expressible as forms of matrix multiplications.

Summary of contributions. As mentioned before, our first result is to show that computing the simulation preorder on an n -state Kripke structure is at least as hard as $n \times n$ boolean matrix multiplication (BMM), explaining why obtaining a “combinatorial” subcubic algorithm seems to be hard [1]. Next, in the case of *acyclic* Kripke structures, we show that fast BMM can be employed to obtain a truly subcubic algorithm, based on a divide-and-conquer technique, and running in $n^{\omega+o(1)}$ -time (where $\omega < 2.376$ is the exponent of matrix multiplication [15]). Since the previous

lower bound also applies to this restricted case, the simulation problem on acyclic structures is essentially equivalent to BMM. Finally, for the acyclic case, we exhibit $O(n^2)$ -size canonical certificates, that can be checked via BMM verification. By transforming BMM into a standard $(+, \times)$ -matrix multiplication, this yields $O(n^2)$ -size certificates that can be verified in $O(n^2)$ time [21, 27, 28].

For cyclic structures, we also provide $O(n^2)$ -size canonical certificates checkable in sub-cubic time; in this case, however, we need to employ a more general variant of matrix multiplication. We introduce the max-semi-boolean matrix multiplication (MSBMM), a multiplication on the semi-ring (\max, \times) , where one of the two matrices contains only zeros and ones. This variant of matrix multiplication is more general than BMM (consider the case where both matrices contain only zeros and ones), but still admits a truly subcubic solution. Indeed, it can be easily transformed into a (\max, \min) -product (where one matrix contains only $+\infty$ and $-\infty$), which in turn can be solved in $n^{(3+\omega)/2+o(1)} \leq O(n^{2.792})$ time [19, 37].

Our last contribution is a reduction from the verification of $n \times n$ MSBMM to the simulation preorder in a $O(n \log n)$ -states cyclic Kripke structure. This is by far the most involved reduction given in this work, and relies on a construction of permutation networks given by Waksman [39] in the '60s (which has applications in the quite distant fields of telecommunications [14, 5, 17] and parallel architectures [16]). This result implies a separation between the acyclic and the cyclic case: our $n^{\omega+o(1)}$ -time lower bound cannot be matched in the acyclic case, unless MSBMM can be also verified in $n^{\omega+o(1)}$ time. (By analogy with the results in [38], we do not expect the verification version to be substantially easier than the computation.) Determining whether this separation holds, i.e., whether max-semi-boolean multiplication is actually harder than boolean multiplication, remains as an open question, sitting among the numerous other problems regarding the complexity of matrix multiplications.

Paper organization. The rest of this paper is organized as follows. In Section 2, we define reachability games and we introduce some notions and classical results about these games, which are useful for this work. In Section 3, we introduce two-tokens reachability games and simulation games, establishing the equivalence between the two (Theorem 9). In Section 4, we present our results for the acyclic case, namely, the reduction from BMM to 2TRGs (Theorem 11), the certificates for 2TRGs with $O(n^2)$ time verification (Theorem 13), and our subcubic divide-and-conquer algorithm (Theorem 16). In Section 5, we present our results for the cyclic case, namely, the certificates verifiable via max-semi-boolean matrix multiplications (Theorem 19) and the reduction from MSBMM verification to 2TRGs (Theorem 20).

2 Reachability games

Definition 1 (Reachability games). A *game-graph* is a structure $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{V}^0, \mathcal{V}^1)$ consisting of a finite set \mathcal{V} of *configurations*, a set $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ of *moves*, and a partition $(\mathcal{V}^0, \mathcal{V}^1)$ of \mathcal{V} into configurations *controlled* respectively by player 0 (Alice) and player 1 (Bob). Configurations and moves form a directed graph $(\mathcal{V}, \mathcal{E})$, called *configuration graph*.

A *play* is a finite or infinite walk in the configuration graph $(\mathcal{V}, \mathcal{E})$, i.e., a non-empty sequence of configurations $\pi = \sigma_0 \sigma_1 \dots \in \mathcal{V}^+ \cup \mathcal{V}^\infty$, such that $(\sigma_i, \sigma_{i+1}) \in \mathcal{E}$ is a move for every two consecutive configurations σ_i and σ_{i+1} in π .

A (positional¹) *strategy* for player $P \in \{0, 1\}$ is a function $s: \mathcal{V}^P \rightarrow \mathcal{V} \cup \{\perp\}$, such that, for every configuration $\sigma \in \mathcal{V}^P$ controlled by P , either $(\sigma, s(\sigma)) \in \mathcal{E}$ is a move, or $s(\sigma) = \perp$. Player P *moves* from σ to $s(\sigma)$ if $(\sigma, s(\sigma)) \in \mathcal{E}$, and *stops* on σ if $s(\sigma) = \perp$. A play $\pi = \sigma_0 \sigma_1 \dots$ is *conforming to* s

¹Non-positional strategies are not needed in this paper. From now on, the term “positional” is omitted.

if, for every configuration σ_i in π , if $\sigma_i \in \mathcal{V}^P$ is controlled by P , then either $(\sigma_i, s(\sigma_i)) \in \mathcal{E}$ and σ_i is followed by $\sigma_{i+1} = s(\sigma_i)$, or $s(\sigma_i) = \perp$ and σ_i is the last configuration of π (i.e. $\pi = \sigma_0 \cdots \sigma_i \in \mathcal{V}^{i+1}$).

A *reachability game* is a pair $(\mathcal{G}, \mathcal{F})$, consisting of a game-graph \mathcal{G} and a partition $\mathcal{F} = (\mathcal{F}^0, \mathcal{F}^1)$ of \mathcal{V} into *winning final configurations* for player 0 and player 1, respectively. A play π is *winning* for player P if it is finite (say, $\pi = \sigma_0 \cdots \sigma_\ell \in \mathcal{V}^{\ell+1}$) and its final configuration is winning² for P (i.e., $\sigma_\ell \in \mathcal{F}^P$). A play is *surviving* for player P if it is either winning for P or infinite. I.e., infinite plays are neither winning nor losing for any player, but they are surviving for both.

A strategy s for player P is *winning* (resp. *surviving*) from $\sigma_0 \subseteq \mathcal{V}$, if every play $\sigma_0 \sigma_1 \cdots$ conforming to s is winning (resp. surviving) for P . The *winning set* $\mathcal{W}^P \subseteq \mathcal{V}$ (resp. the *surviving set* $\mathcal{S}^P \subseteq \mathcal{V}$) of P is the set of configurations from which P has a winning (resp. surviving) strategy.

Definition 2 (Set lifting operator, closed set). Let $\mathcal{U} \subseteq \mathcal{V}$. The set $\text{LiftSet}^P(\mathcal{U}) \subseteq \mathcal{V}$ contains all the configurations from which player P can be sure to either win immediately, or that at the next turn the game will move to a configuration in \mathcal{U} . It is defined as follows:

$$\sigma \in \text{LiftSet}^P(\mathcal{U}) \iff \begin{cases} \sigma \in \mathcal{F}^P \vee \bigvee_{(\sigma, \sigma') \in \mathcal{E}} \sigma' \in \mathcal{U} & \text{if } \sigma \in \mathcal{V}^P \\ \sigma \in \mathcal{F}^P \wedge \bigwedge_{(\sigma, \sigma') \in \mathcal{E}} \sigma' \in \mathcal{U} & \text{if } \sigma \in \mathcal{V}^{1-P} \end{cases}$$

where as usual $\bigvee_{x \in \emptyset} := \text{false}$ and $\bigwedge_{x \in \emptyset} := \text{true}$. A set $\mathcal{U} \subseteq \mathcal{V}$ is *closed* for P if $\mathcal{U} \subseteq \text{LiftSet}^P(\mathcal{U})$.

Definition 3 (Potential, lifting operator, progress measure). A *potential* is a function $p: \mathcal{V} \rightarrow \mathbb{N} \cup \{\infty\}$. Let $\text{supp}(p) = \{\sigma \in \mathcal{V} \mid p(\sigma) < \infty\}$. We define the potential $\text{Lift}^P(p)$ as follows

$$\text{Lift}^P(p)(\sigma) = \begin{cases} \min \{p_\perp^P(\sigma)\} \cup \{1 + p(\sigma') \mid (\sigma, \sigma') \in \mathcal{E}\} & \text{if } \sigma \in \mathcal{V}^P \\ \max \{p_\perp^P(\sigma)\} \cup \{1 + p(\sigma') \mid (\sigma, \sigma') \in \mathcal{E}\} & \text{if } \sigma \in \mathcal{V}^{1-P} \end{cases}$$

where $p_\perp^P(\sigma) = 0$ if $\sigma \in \mathcal{F}^P$, $p_\perp^P(\sigma) = \infty$ if $\sigma \in \mathcal{F}^{1-P}$, and $1 + \infty := \infty$. A potential p is a *progress measure* for P if $p(\sigma) \geq \text{Lift}^P(p)(\sigma)$ for every $\sigma \in \mathcal{V}$.

Proposition 4 (Characterizations of reachability games). *The following properties hold:*

- (a) if $\mathcal{U} \subseteq \mathcal{V}$ is closed for P then there is a strategy for P surviving from every $\sigma \in \mathcal{U}$,
- (b) if p is a progress measure for P then there is a strategy for P winning from every $\sigma \in \text{supp}(p)$,
- (c) \mathcal{W}^P and \mathcal{S}^P are respectively the least and the greatest fixpoints of LiftSet^P ,
- (d) the operator Lift^P has a unique fixpoint r^P and we have $\mathcal{W}^P = \text{supp}(r^P)$ and $\mathcal{S}^{1-P} = \mathcal{V} \setminus \text{supp}(r^P)$,
- (e) this fixpoint r^P can be computed in linear time $O(|\mathcal{V}| + |\mathcal{E}|)$.

Proof. Properties (a)–(d) are variants of classical results in infinite two-players games [26, 20, 11], and applications of the Tarski theorem. As for property (e), the measure r^P can be computed by an alternating backward search [2, 4]. A rigorous proof of all the properties is given in Appendix A. \square

²In the definition of reachability games usually found in the literature, it is sufficient that a configuration in a certain set F is reached, anywhere in a play, and the play is considered winning for player 0. In our definition, a play must be explicitly *stopped* on a configuration in \mathcal{F}^P , in order to be winning for player P . In particular, if the play reaches a configuration $\sigma \in \mathcal{F}^P \cap \mathcal{V}^P$, then player P can stop at σ and win immediately, but if $\sigma \in \mathcal{F}^P \cap \mathcal{V}^{1-P}$, then player $1 - P$ may choose a next move (if there is any) and the game continues. This difference makes the description of our results simpler; nevertheless, it is easy to establish an equivalence between the two variants.

3 Two-tokens and simulation games

Definition 5 (Two-tokens reachability games). A *two-tokens game-graph* over two finite directed graphs $G_0 = (V_0, E_0)$ and $G_1 = (V_1, E_1)$, is a game-graph $\mathcal{G}(G_0, G_1)$ defined as follows. For every player $P \in \{0, 1\}$, and every pair of vertices $u \in V_P$ and $v \in V_{1-P}$, there is a configuration $\langle P, u, v \rangle$, controlled by P . In the configuration $\langle P, u, v \rangle$, a token belonging to P is located on $u \in V_P$, and a token belonging to $1 - P$ is located on $v \in V_{1-P}$. The player holding the turn can move her token along any edge $(u, u') \in E_P$ of her graph G_P , and then pass the turn to the other player, resulting in the move $(\langle P, u, v \rangle, \langle 1 - P, v, u' \rangle)$. (Observe that, to keep symmetry between the two players, u' and v are swapped.) Summarizing, $\mathcal{G}(G_0, G_1) = \mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{V}^0, \mathcal{V}^1)$, where $\mathcal{V}^P = \{\langle P, u, v \rangle \mid u \in V_P, v \in V_{1-P}\}$, $\mathcal{V} = \mathcal{V}^0 \cup \mathcal{V}^1$, and $\mathcal{E} = \{(\langle P, u, v \rangle, \langle 1 - P, v, u' \rangle) \mid P \in \{0, 1\}, (u, u') \in E_P, v \in V_{1-P}\}$.

A *two-tokens reachability game* (2TRG) is a reachability game $(\mathcal{G}, \mathcal{F})$ with $\mathcal{G} = \mathcal{G}(G_0, G_1)$. The problem 2TRG WINNING SET (2TRG-WS) of order $|V_0| \times |V_1|$ asks to compute its winning and survival sets, given the graphs G_0, G_1 and the partition \mathcal{F} . In the ACYCLIC variant, G_0 and G_1 are acyclic. In the SEMI-ACYCLIC variant, at least one among G_0 and G_1 is acyclic.

Definition 6 (Kripke structure). A *Kripke structure* is a structure $\mathcal{K} = (S, T, L)$ consisting of a set of states S , a transition relation $T \subseteq S \times S$, and a labeling function $L: S \rightarrow \Lambda$ over the states.³

Definition 7 (Simulation game). A *simulation game* for a Kripke structure $\mathcal{K} = (S, T, L)$ is a 2TRG $(\mathcal{G}, \mathcal{F})$, where the two-tokens game-graph $\mathcal{G} = \mathcal{G}(G, G)$ is built over two copies of the graph $G = (S, T)$, and $\mathcal{F}^1 = \{\langle 0, s, t \rangle \in \mathcal{V}^0 \mid L(s) = L(t)\}$. A state $t \in S$ *simulates* a state $s \in S$ (written $s \preceq_s t$) if $\langle 0, s, t \rangle \in \mathcal{F}^1$. The relation \preceq_s is called⁴ *simulation preorder*.

Computing the relation \preceq_s over $S \times S$ is an instance of the problem SIMULATION of order $|S|$. In the ACYCLIC variant of SIMULATION, the graph (S, T) is required to be acyclic.

Remark 8. The simulation preorder can be computed in $O(nm)$ time (assuming $m \geq n$) by solving $(\mathcal{G}, \mathcal{F})$ as in Proposition 4 (e), since $|\mathcal{V}| = O(n^2)$, $|\mathcal{E}| = O(nm)$, and \mathcal{G} can be constructed efficiently. We point out that the classical $O(mn)$ time algorithms for simulation citeHenzinger1995,Bloom1995 can be regarded as more elaborated instantiations of this algorithm.

Theorem 9 (2TRGs and simulation games are equivalent). *Given any (acyclic) 2TRG $(\mathcal{G}, \mathcal{F})$ of order $n_0 \times n_1$, there exist an (acyclic) simulation game $(\mathcal{G}', \mathcal{F}')$, on a Kripke structure $\mathcal{K} = (S, T, L)$, and a map⁵ $f: \mathcal{V}_{\mathcal{G}} \rightarrow \mathcal{V}_{\mathcal{G}'}$, such that $|S| = n = O(n_0 + n_1)$, the structure \mathcal{K} is constructible in $O(n^2)$ time, f is computable in $O(1)$ time, and $\langle P, u, v \rangle \in \mathcal{S}_{\mathcal{G}, \mathcal{F}}^1 \iff f(\langle P, u, v \rangle) \in \mathcal{S}_{\mathcal{G}', \mathcal{F}'}^1$.*

Proof. Let $S = V_0 \cup V_0^* \cup V_1$ where $V_0^* = \{u^* \mid u \in V_0\}$ (assuming unions are disjoint). Label each state $u^* \in V_0^*$ with a distinct label $L(u^*) = \lambda_u$, and all the other states $x \in V_0 \cup V_1$ with the same label $L(x) = \lambda_{\square}$. Let $T = E_0 \cup E_1 \cup \{(u, u^*) \mid u \in V_0\} \cup \{(v, u) \mid \langle 1, v, u \rangle \in \mathcal{F}^1\} \cup \{(v, u^*) \mid \langle 0, u, v \rangle \in \mathcal{F}^1\}$, and let $f: \mathcal{V}_{\mathcal{G}} \rightarrow \mathcal{V}_{\mathcal{G}'}$ be the inclusion.

(\implies) Bob survives in $(\mathcal{G}', \mathcal{F}')$ with the same strategy as in $(\mathcal{G}, \mathcal{F})$, unless one of the following occurs: (a) the strategy of Bob says to stop and win on a configuration $\langle 1, v, u \rangle \in \mathcal{F}^1$, or (b) Alice moves to a starred node u^* , say, from $\langle 0, u, v \rangle$ to $\langle 1, v, u^* \rangle$. In case (a), instead of stopping, take the edge (v, u) given by construction. In case (b) take the edge (v, u^*) , which is present since

³The transition relation T is sometimes required to be left-total (i.e. $\forall s \in S \exists s' \in S$ such that $(s, s') \in T$), and the label universe Λ is usually defined as the power set of a given set of atomic proposition. These requirements are not relevant to our discussion, and have been omitted. According to the definition above, a Kripke structure is nothing more than a vertex-labeled directed graph.

⁴The equivalence between this definition of simulation preorder and a more classical one is given in Appendix B.

⁵We use subscripts to distinguish between objects associated with $(\mathcal{G}, \mathcal{F})$ and with $(\mathcal{G}', \mathcal{F}')$.

$\langle 0, u, v \rangle \in \mathcal{F}^1$ (otherwise Alice would have won in $(\mathcal{G}, \mathcal{F})$ by stopping on $\langle 0, u, v \rangle$). In both cases (a) and (b), Bob moves to a configuration $\langle 0, x, x \rangle$ where the two tokens are located on the same vertex. From now on, Bob can copy all the moves of Alice and survive forever.

(\Leftarrow) Suppose that Alice wins in $(\mathcal{G}, \mathcal{F})$ from $\langle P, u, v \rangle$. She survives in $(\mathcal{G}', \mathcal{F}')$ applying the same strategy, until one of the following occurs: (a) the strategy says to stop on some winning final configuration $\langle 0, u, v \rangle \in \mathcal{F}^0$, or (b) Bob moves from $\langle 1, v, u \rangle$ to $\langle 0, u, x \rangle$ with $x \in V_0 \cup V_0^*$. In both cases, take the edge (u, u^*) . We show that Bob cannot move to u^* in the next turn, and since u^* is the only state labeled with λ_u , Alice wins. In case (a), there is no edge (v, u^*) since $\langle 0, u, v \rangle \in \mathcal{F}^0$. In case (b), we have $\langle 1, v, u \rangle \in \mathcal{F}^0$ (otherwise Bob could have won in $(\mathcal{G}, \mathcal{F})$ by stopping on $\langle 1, v, u \rangle$), so $x \neq u$ and there exists no edge (x, u^*) . (See Appendix C for a more formal proof.) \square

4 Acyclic case

We start by showing that 2TRGs are at least as hard as boolean matrix multiplication.

Definition 10 (Boolean matrix multiplication). Given an $n_1 \times n_2$ boolean matrix \mathbf{B}_1 and an $n_2 \times n_3$ boolean matrix \mathbf{B}_2 , their boolean product is the $n_1 \times n_3$ boolean matrix $\mathbf{B}_1 \star \mathbf{B}_2$ defined by: $(\mathbf{B}_1 \star \mathbf{B}_2)[i, j] = \bigvee_{k=1}^{n_2} \mathbf{B}_1[i, k] \wedge \mathbf{B}_2[k, j]$. The problem BOOLEAN MATRIX MULTIPLICATION (BMM) of size $n_1 \times n_2 \times n_3$ asks to compute $\mathbf{B}_1 \star \mathbf{B}_2$ given \mathbf{B}_1 and \mathbf{B}_2 .

Theorem 11. *If 2TRG WINNING SET of order $n \times n$ can be solved in $O(n^\alpha)$ time, for some $\alpha \geq 2$, then BOOLEAN MATRIX MULTIPLICATION of size $n \times n \times n$ can be computed in $O(n^\alpha)$ time.*

Proof. We want to compute the boolean product $\mathbf{B}_1 \star \mathbf{B}_2$ between two $n \times n$ matrices. Consider a 2TRG where $V_0 = \{x_1, \dots, x_n\} \cup \{y_1, \dots, y_n\}$ and $V_1 = \{z_1, \dots, z_n\}$. The edges of G_0 go only from nodes of type x to nodes of type y , and are defined using \mathbf{B}_1 , namely $E_0 = \{(x_i, y_k) \mid \mathbf{B}_1[i, k] = 1\}$. The graph G_1 has no edges. The matrix \mathbf{B}_2 is used to define the goal configurations for Alice $\mathcal{F}^0 = \{\langle 1, z_j, y_k \rangle \mid \mathbf{B}_2[k, j] = 1\}$.

Suppose the play starts from $\langle 0, x_i, z_j \rangle$. Since Bob has no moves, the play lasts at most one turn. Hence, the only way for Alice to win is to reach in a single move a winning final configuration $\langle 1, z_j, y_k \rangle \in \mathcal{F}^0$. This is possible iff, for some $k \in \{1, \dots, n\}$, we have both $(x_i, y_k) \in E_0$ and $\langle 1, z_j, y_k \rangle \in \mathcal{F}^0$, i.e. both $\mathbf{B}_1[i, k] = 1$ and $\mathbf{B}_2[k, j] = 1$. That is, we have $\langle 0, x_i, z_j \rangle \in \mathcal{W}^0 \iff (\mathbf{B}_1 \star \mathbf{B}_2)[i, j] = 1$, so by computing the winning set \mathcal{W}^0 we compute the product $\mathbf{B}_1 \star \mathbf{B}_2$. \square

The following lemma is used both for verification (Theorem 13) and, later, for our divide-and-conquer algorithm.

Lemma 12 (Computing *LiftSet* via BMM). *The operator LiftSet^P in 2TRGs of order at most $n \times n$ can be computed (verified) by computing (verifying) two boolean matrix multiplications of size at most $n \times n \times n$, with only $O(n^2)$ extra time.*

Proof. Write $V_P = \{u_1, \dots, u_{n_P}\}$ and $V_{1-P} = \{v_1, \dots, v_{n_{1-P}}\}$. Given a configuration $\langle P, u_i, v_j \rangle \in \mathcal{V}^P$, where player P holds the turn, we have by definition $\langle P, u_i, v_j \rangle \in \text{LiftSet}^P(\mathcal{U})$ iff she can either win immediately ($\langle P, u_i, v_j \rangle \in \mathcal{F}^P$) or she can move to some configuration $\langle 1-P, v_j, u_k \rangle \in \mathcal{U}$. This holds iff, for some $k \in \{1, \dots, n_P\}$, we have both $(u_i, u_k) \in E_P$ and $(1-P, v_j, u_k) \in \mathcal{U}$. By considering the adjacency matrix \mathbf{E} of G_P (i.e., a $n_P \times n_P$ boolean matrix with $\mathbf{E}[i, j] = 1$ iff $(u_i, u_j) \in E_P$), and the $n_P \times n_{1-P}$ boolean matrix \mathbf{U} where $\mathbf{U}[i, j] = 1$ iff $(1-P, v_j, u_i) \in \mathcal{U}$, this is equivalent to saying that $(\mathbf{E} \star \mathbf{U})[i, j] = 1$. In formulas:

$$\langle P, u_i, v_j \rangle \in \text{LiftSet}^P(\mathcal{U}) \iff \langle P, u_i, v_j \rangle \in \mathcal{F}^P \vee (\mathbf{E} \star \mathbf{U})[i, j].$$

For those configurations $\langle 1 - P, v_j, u_i \rangle \in \mathcal{V}^{1-P}$, where player P does not hold the turn, we can work by dualization. Indeed, by applying the de Morgan law $\langle 1 - P, v_j, u_i \rangle \in \text{LiftSet}^P(\mathcal{U}) \iff \langle 1 - P, v_j, u_i \rangle \notin \text{LiftSet}^{1-P}(\mathcal{V} \setminus \mathcal{U})$, we reduce ourselves to the previous case, where P is substituted with $1 - P$ and \mathcal{U} with $\mathcal{V} \setminus \mathcal{U}$. Summarizing, one BMM is needed to compute $\text{LiftSet}^P(\mathcal{U}) \cap \mathcal{V}^P$ and a second BMM is needed to compute $\text{LiftSet}^P(\mathcal{U}) \cap \mathcal{V}^{1-P}$, by dualization. It is clear that we spend no more than $O(n^2)$ time, besides the computation of the two BMMs. \square

Theorem 13. (SEMI-)ACYCLIC 2TRG-WS of order $n \times n$ can be verified via boolean matrix product verification of size $n \times n \times n$, with only $O(n^2)$ extra time.

ACYCLIC SIMULATION of order n and (SEMI-)ACYCLIC 2TRG-WS of order $n \times n$ admit a $O(n^2)$ -size certificate that can be verified via standard $(+, \times)$ -matrix product verification of size $n \times n \times n$.

Proof. If any of G_0 and G_1 is acyclic, then $\mathcal{G} = \mathcal{G}(G_0, G_1)$ is also acyclic and there are no infinite plays. Hence, the set $\mathcal{S}^P = \mathcal{W}^P$ is the unique solution of the equation $\text{LiftSet}^P(\mathcal{U}) = \mathcal{U}$. To verify that $\mathcal{U} = \mathcal{S}^P$ for a given set $\mathcal{U} \subseteq \mathcal{V}$, it is sufficient to verify that this equation holds, which, by Lemma 12, is equivalent to verifying two BMMs. The result of the standard $(+, \times)$ -matrix multiplications corresponding to these two BMMs can be used as a certificate, so that they be verified in $O(n^2)$ time. \square

To describe our algorithm for acyclic 2TRGs (Theorem 16), we first present our approach on reachability games.

Definition 14 (Induced sub-game-graph and sub-partition). Let $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{V}^0, \mathcal{V}^1)$ be a game-graph and $\mathcal{U} \subseteq \mathcal{V}$. The *sub-game-graph* of \mathcal{G} induced by \mathcal{U} is the game-graph $\mathcal{G}[\mathcal{U}] = (\mathcal{U}, \mathcal{E} \cap (\mathcal{U} \times \mathcal{U}), \mathcal{V}^0 \cap \mathcal{U}, \mathcal{V}^1 \cap \mathcal{U})$. Given a partition $\mathcal{F} = (\mathcal{F}^0, \mathcal{F}^1)$ of \mathcal{V} , the *sub-partition* $\mathcal{F}[\mathcal{U}]$ induced by \mathcal{U} is $(\mathcal{F}^0 \cap \mathcal{U}, \mathcal{F}^1 \cap \mathcal{U})$.

Lemma 15 (Dicut decomposition of reachability games). Let $(\mathcal{V}_T, \mathcal{V}_H)$ be a dicut of the configuration graph $(\mathcal{V}, \mathcal{E})$, i.e., a partition $(\mathcal{V}_T, \mathcal{V}_H)$ of \mathcal{V} such that there are no edges from \mathcal{V}_H to \mathcal{V}_T . Define the following:

- (1) \mathcal{S}_H^P is the surviving set of P in the game $(\mathcal{G}[\mathcal{V}_H], \mathcal{F}[\mathcal{V}_H])$,
- (2) $\mathcal{F}_* = (\mathcal{F}_*^0, \mathcal{F}_*^1)$ is a partition of \mathcal{V} with $\mathcal{F}_*^P = \text{LiftSet}^P(\mathcal{S}_H^P)$,
- (3) \mathcal{S}_T^P is the surviving set of P in the game $(\mathcal{G}[\mathcal{V}_T], \mathcal{F}_*[\mathcal{V}_T])$.

Then, the surviving set \mathcal{S}^P in the original game can be written as $\mathcal{S}^P = \mathcal{S}_H^P \cup \mathcal{S}_T^P$.

Proof. Fix $P = 0$. If a play starts with a configuration $\sigma_0 \in \mathcal{V}_H$, then it never reaches any configuration in \mathcal{V}_T , since there are no backward edges in the dicut. Hence, for all the initial configurations in \mathcal{V}_H , the problem is equivalent in the sub-game $(\mathcal{G}[\mathcal{V}_H], \mathcal{F}[\mathcal{V}_H])$, i.e. $\mathcal{S}^0 \cap \mathcal{V}_H = \mathcal{S}_H^0$.

Consider now a configuration $\sigma \in \mathcal{V}^0$ where Alice holds the turn. If she can move to a configuration $\sigma' \in \mathcal{S}_H^0$, which is surviving, then we can assume that she will take this opportunity and survive. Hence, σ can be added to the final winning configurations of Alice, and all the outgoing moves from σ can be removed. Indeed, after this change, σ will still be a surviving configuration (actually, winning) for Alice. Now take a configuration $\sigma \in \mathcal{V}^1$ where Bob holds the turn. If he has no other choice but to either stop, and lose immediately, or move to a configuration $\sigma' \in \mathcal{S}_H^0$, surviving for his opponent Alice, then he clearly cannot win from σ . Hence, also in this case, σ can be added to the final winning configurations of Alice and all the outgoing moves removed.

In general, the new set \mathcal{F}_*^0 of winning final configurations for Alice can be defined as $\mathcal{F}_*^0 = \text{LiftSet}^0(\mathcal{S}_H^0)$, and all the moves from \mathcal{V}_T to \mathcal{V}_H can be removed. To solve the problem for the second part of the game-graph, we can now work on the sub-game-graph $\mathcal{G}[\mathcal{V}_T]$, but only after replacing the winning final configurations with the new partition $\mathcal{F}_*[\mathcal{V}_T]$. We obtain $\mathcal{S}^0 \cap \mathcal{V}_T = \mathcal{S}_T^0$ and the statement of the lemma follows. (See Appendix D for a more formal proof.) \square

Theorem 16. *ACYCLIC SIMULATION and ACYCLIC 2TRG WINNING SET can be computed in $n^{\omega+o(1)}$ time, for any ω such that boolean matrix multiplication can be solved in $n^{\omega+o(1)}$ time.*

Proof. Let (V_0^T, V_0^H) be a dicut of G_0 (i.e., a partition of V such that $E \cap (V_0^H \times V_0^T) = \emptyset$) with $|V_0^T|, |V_0^H| \leq \lceil n/2 \rceil$. Such a dicut can be easily obtained from a topological sort of G_0 , splitting at about half. Observe that the dicut (V_0^T, V_0^H) induces a dicut $(\mathcal{V}_T, \mathcal{V}_H)$ of the configuration graph of $\mathcal{G} = \mathcal{G}(G_0, G_1)$, with $\mathcal{G}[\mathcal{V}_X] = \mathcal{G}(G_0[V_0^X], G_1)$ for $X \in \{T, H\}$. To compute \mathcal{S}^0 , we apply the formula $\mathcal{S}^0 = \mathcal{S}_H^0 \cup \mathcal{S}_T^0$ given by Lemma 15, where \mathcal{S}_H^0 and \mathcal{S}_T^0 are computed recursively and $\mathcal{F}_*^0 = \text{LiftSet}^0(\mathcal{S}_H^0)$ is computed via fast BMM in $n^{\omega+o(1)}$ time (by Lemma 12). Crucially, at each recursive call we dualize the game, swapping the two players. The running time $T(n_0, n_1)$ then satisfies the recurrence $T(n_0, n_1) \leq 2T(n_1, \lceil n_0/2 \rceil) + (n_0 + n_1)^{\omega+o(1)} \leq 4T(\lceil n_0/2 \rceil, \lceil n_1/2 \rceil) + (n_0 + n_1)^{\omega+o(1)}$. Under the assumption $\omega \geq 2$, we get $T(n, n) \leq n^{\omega+o(1)}$. (If $\omega = 2$, the extra logarithmic factor is accounted for in the $n^{o(1)}$ term.) \square

5 Cyclic case

We first show how max-semi-boolean matrix multiplication can be employed for the verification of 2TRG WINNING SET (Lemma 19).

Definition 17 (min-/max-semi-boolean matrix multiplication). Given an $n_1 \times n_2$ matrix of numbers⁶ \mathbf{A} and an $n_2 \times n_3$ boolean matrix \mathbf{B} , their min- and max-semi-boolean products are the $n_1 \times n_3$ matrices $\mathbf{A} \star_{\min} \mathbf{B}$ and $\mathbf{A} \star_{\max} \mathbf{B}$ defined as follows:

$$\begin{aligned} (\mathbf{A} \star_{\min} \mathbf{B})[i, j] &= \min \{ \mathbf{A}[i, k] \mid k = 1, \dots, n_2 \text{ and } \mathbf{B}[k, j] = 1 \} \\ (\mathbf{A} \star_{\max} \mathbf{B})[i, j] &= \max \{ \mathbf{A}[i, k] \mid k = 1, \dots, n_2 \text{ and } \mathbf{B}[k, j] = 1 \}. \end{aligned}$$

The problems MIN- and MAX-SEMI-BOOLEAN MATRIX MULTIPLICATION (MSBMM) of size $n_1 \times n_2 \times n_3$ ask to compute $\mathbf{A} \star_{\min} \mathbf{B}$ and $\mathbf{A} \star_{\max} \mathbf{B}$ given \mathbf{A} and \mathbf{B} . In the DISTINCT variant of MSBMM, we require $\mathbf{A}[i, k] \neq \mathbf{A}[i, k']$ for $k \neq k'$.

The min and max versions are clearly equivalent since $\mathbf{A} \star_{\min} \mathbf{B} = -((-\mathbf{A}) \star_{\max} \mathbf{B})$. Observe that, given a MSBMM, we can replace $\mathbf{A}[i, k]$ with its rank in the set $\{ \mathbf{A}[i, k] \mid k = 1, \dots, n_2 \}$, breaking ties arbitrarily, and we get an equivalent DISTINCT MSBMM problem.

Lemma 18 (Computing *Lift* via MSBMM). *The operator Lift in 2TRGs of order at most $n \times n$ can be computed (verified) by computing (verifying) two MSBMM of size at most $n \times n \times n$, with only $O(n^2)$ extra time.*

Proof. Write $V_P = \{u_1, \dots, u_{n_0}\}$ and $V_{1-P} = \{v_1, \dots, v_{n_1}\}$. Consider the adjacency matrix \mathbf{E} of G_P (an $n_P \times n_P$ boolean matrix with $\mathbf{E}_P[i, j] = 1$ iff $(u_i, u_j) \in E_P$), and define the $n_P \times n_{1-P}$ matrix \mathbf{P} with $\mathbf{P}[i, j] = p(\langle 1 - P, v_j, u_i \rangle)$. For a configuration $\langle P, u_i, v_j \rangle \in \mathcal{V}^P$, where player P holds the turn, we have $\text{Lift}^P(p)(\langle P, u_i, v_j \rangle) = 0$ if $\langle P, u_i, v_j \rangle \in \mathcal{F}^P$, and otherwise

$$\text{Lift}^P(p)(\langle P, u_i, v_j \rangle) = \min_{(u_i, u_k) \in E_P} p(\langle 1 - P, v_j, u_k \rangle) = \min \{ \mathbf{P}[k, j] \mid \mathbf{E}[i, k] = 1 \} = (\mathbf{P} \star_{\min} \mathbf{E})[i, j].$$

⁶Integers, reals, or elements of any totally ordered set.

To compute $Lift^P(p)$ for those configurations in which the other player $1 - P$ holds the turn, we dualize the problem: we show equivalently how to compute $Lift^{1-P}(p)(\langle P, u_i, v_j \rangle)$ for $\langle P, u_i, v_j \rangle \in \mathcal{V}^P$. Similarly as in the previous case, we obtain $Lift^{1-P}(p)(\langle P, u_i, v_j \rangle) = \infty$ if $\langle P, u_i, v_j \rangle \in \mathcal{F}^{1-P}$, and $Lift^{1-P}(p)(\langle P, u_i, v_j \rangle) = (\mathbf{P} \star_{\max} \mathbf{E})[i, j]$ otherwise.

Hence, a total of two MSBMMs are needed to compute the potential $Lift^P(p)$: one for the configurations in \mathcal{V}^P , and another for the configurations in \mathcal{V}^{1-P} , after dualization. \square

Theorem 19. SIMULATION of order n and 2TRG WINNING SET of order $n \times n$ admit $O(n^2)$ -size canonical certificates that can be checked by verifying two MAX-SEMI-BOOLEAN MATRIX MULTIPLICATIONS of size $n \times n \times n$, and only $O(n^2)$ extra time.

Proof. Recall that r^P is the only solution of the equation $r^P = Lift^P(r^P)$, and that $\mathcal{W}^P = \text{supp}(r^P)$. Hence, r^P is a $O(n^2)$ -size certificate, and it can be checked by verifying the fixpoint equation using two MSBMMs (by Lemma 18). \square

The rest of this section is devoted to proving the following theorem.

Theorem 20. The verification of DISTINCT MAX-SEMI-BOOLEAN MATRIX MULTIPLICATION of size $n \times m \times m$ can be reduced to the verification of 2TRG WINNING SET of order $n \times m \log m$.

Corollary 21. If SIMULATION of order n can be computed or verified in $O(n^\alpha)$ time for $\alpha \geq 2$, then DISTINCT MSBMM of size $n \times n \times n$ can be verified in $O(n^\alpha \log n)$ time.

An $m \times m$ boolean matrix \mathbf{B} and two $n \times m$ matrices of numbers \mathbf{A} and \mathbf{C} are given, where $\mathbf{A}[i, k] \neq \mathbf{A}[i, k']$ for $k \neq k'$. We want to check whether $\mathbf{C}[i, j] = (\mathbf{A} \star_{\max} \mathbf{B})[i, j]$ for every i and j . Fixed $i \in \{1, \dots, n\}$ and $j \in \{1, \dots, m\}$, let k_{ij} be the only index such that $\mathbf{A}[i, k_{ij}] = \mathbf{C}[i, j]$. If there is no such k_{ij} , or $\mathbf{B}[k_{ij}, j] = 0$, then clearly the answer is no. Otherwise, $\mathbf{C}[i, j] \leq (\mathbf{A} \star_{\max} \mathbf{B})[i, j]$ for every i, j . It remains to check that there is no triple (i, j, k) such that $\mathbf{A}[i, k] > \mathbf{C}[i, j]$ with $\mathbf{B}[k, j] = 1$. We call such a triple an *invalid triangle*. We construct in $O(nm \log m)$ time a 2TRG $(\mathcal{G}(G_0, G_1), \mathcal{F})$, where Bob survives on some initial configurations iff there exists an invalid triangle, and we conclude by checking that Alice wins from every configuration.

The graph G_0 contains n isolated loops, i.e., $V_0 = \{1, \dots, n\}$ and $E_0 = \{(i, i) \mid i \in V_0\}$, so that the token of Alice always remains in its initial position $i \in V_0$. The graph G_1 is built in such a way that, if (and only if) there is an invalid triangle (i, j, k) , then Bob can move his token in a cycle, without encountering losing configurations, and survive. Since the graph G_1 cannot depend on i , to achieve this goal we can only manipulate the winning final configurations. Moreover, we need to keep the number of vertices low to $O(m \log m)$. To this end, it comes to help the construction of permutation networks.

A *permutation network* [39] of size n is defined as follows. There are n *inlets* u_1, \dots, u_n and n *outlets* v_1, \dots, v_n . Between the inlets and the outlets, there is a set of *gates* S , and for each gate $s \in S$ there are two input ports x_1^s, x_2^s and two output ports y_1^s, y_2^s . A gate connects each of the two input ports to an output port: when the gate is active, they are crossed and swapped, otherwise they are connected in order. Inlets, gate ports and outlets are connected with *wires*, which form a bijective relation $W \subseteq O \times I$ between $O = \{u_1, \dots, u_n\} \cup \bigcup_{s \in S} \{y_1^s, y_2^s\}$ and $I = \bigcup_{s \in S} \{x_1^s, x_2^s\} \cup \{v_1, \dots, v_n\}$. The property of the network is as follows: for every permutation $\pi: \{1, \dots, n\} \rightarrow \{1, \dots, n\}$, there is a subset of the gates $A_\pi \subseteq S$ which have to be activated, so that the network realizes the permutation π between the inlets and the outlets.⁷ Waksman [39] shows a

⁷Given π , define the directed graph $G_\pi = (I \cup O, W \cup T_\pi)$, where $T_\pi \subseteq I \times O$ contains all the pairs of the form (x_i^s, y_j^s) , for $s \in S$ and $i, j \in \{1, 2\}$, with $i \neq j$ if $s \in A_\pi$ and $i = j$ otherwise. For every permutation π , the graph G_π is the union of n vertex-disjoint paths P_1, \dots, P_n , where P_i goes from u_i to $v_{\pi(i)}$.

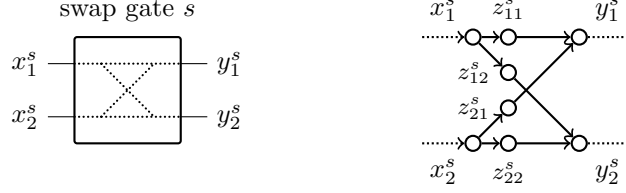


Figure 1: Visual representation of a swap gate and its corresponding gate gadget graph.

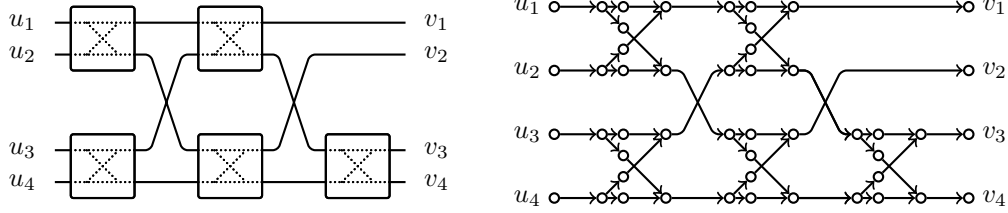


Figure 2: A permutation network of size $n = 4$ and its corresponding permutation gadget graph.

construction of permutation networks of size $n = 2^k$ where $|S| = O(n \log n)$ and A_π is computable in $O(n \log n)$ time for every π .

To realize a permutation network in the graph G_1 , we have to show how to implement a gate. We define the *gate gadget* graph (Fig. 1) as follows: there are two input vertices x_1^s and x_2^s and two output vertices y_1^s and y_2^s , corresponding to the ports of the gate $s \in S$, then there are four guard vertices z_{jk}^s and eight edges $x_j^s \rightarrow z_{jk}^s$ and $z_{jk}^s \rightarrow y_k^s$, for $j, k \in \{1, 2\}$. By removing the guard vertices z_{12} and z_{21} , this gadget behaves like an inactive gate, while by removing z_{11} and z_{22} it behaves like an active gate. The *permutation gadget* graph X of size n (Fig. 2) contains the inlets u_1, \dots, u_n and the outlets v_1, \dots, v_n as vertices, a gate gadget for each gate $s \in S$, and all the wires W as extra edges. For a given permutation π , define $K_s^X(\pi) = \{z_{11}^s, z_{22}^s\}$ for $s \in A_\pi$, $K_s^X(\pi) = \{z_{12}^s, z_{21}^s\}$ for $s \in S \setminus A_\pi$ and $K^X(\pi) = \bigcup_{s \in S} K_s^X(\pi)$. By removing all the vertices in $K^X(\pi)$, the gadget realizes the permutation π : the only maximal paths in the graph not passing through $K^X(\pi)$ are P_1, \dots, P_n , where P_i goes from u_i to $v_{\pi(i)}$. This completes the construction of the gadget.

We now define a family of permutations π_i indexed by $i \in \{1, \dots, n\}$. Identify the k -th columns of \mathbf{A} with the index $\ell = k$ and the j -th column of \mathbf{C} with the index $\ell = m + j$. For every i , let $\pi_i: \{1, \dots, 2m\} \rightarrow \{1, \dots, 2m\}$ be a permutation that sorts the indexes $\ell \in \{1, \dots, 2m\}$ according to the value $\mathbf{A}[i, k]$ for $\ell = k \in \{1, \dots, m\}$ and $\mathbf{C}[i, j]$ for $\ell = m + j \in \{m + 1, \dots, 2m\}$, breaking ties in favor of \mathbf{A} . Namely, π_i is such that $\mathbf{A}[i, k] > \mathbf{C}[i, j]$ implies $\pi_i(k) > \pi_i(m + j)$ and $\mathbf{A}[i, k] \leq \mathbf{C}[i, j]$ implies $\pi_i(k) < \pi_i(m + j)$.

The graph G_1 is constructed as follows (see Fig. 3). We start with four distinct vertices $x_\ell, y_\ell, z_\ell, w_\ell$ for each $\ell \in \{1, \dots, 2m\}$, forming four successive layers. Between the first two layers, we add a permutation gadget Y of size $2m$, with inlets x_1, \dots, x_{2m} and outlets y_1, \dots, y_{2m} , associated with the permutation family $\pi_i^Y = \pi_i$. Then, we add an edge $(y_\ell, z_{\ell'})$ for each $\ell, \ell' \in \{1, \dots, 2m\}$ with $\ell' \leq \ell$. Between the third and the fourth layers, we add a permutation gadget Z of size $2m$ with inlets z_1, \dots, z_{2m} and outlets w_1, \dots, w_{2m} , associated with the family of inverse permutations $\pi_i^Z = \pi_i^{-1}$. Finally, for each $j, k \in \{1, \dots, m\}$ such that $\mathbf{B}[k, j] = 1$, we add a “return” edge (w_{m+j}, x_k) connecting the last with the first layer. The set of final winning configurations for Alice is $\mathcal{F}^0 = \mathcal{V}^1 \cup \{(0, i, u) \mid i \in \{0, \dots, n\} \text{ and } u \in K^Y(\pi_i) \cup K^Z(\pi_i^{-1})\}$.

Our reduction is complete. We conclude the proof of Theorem 20 with Lemma 22 and Lemma 23.

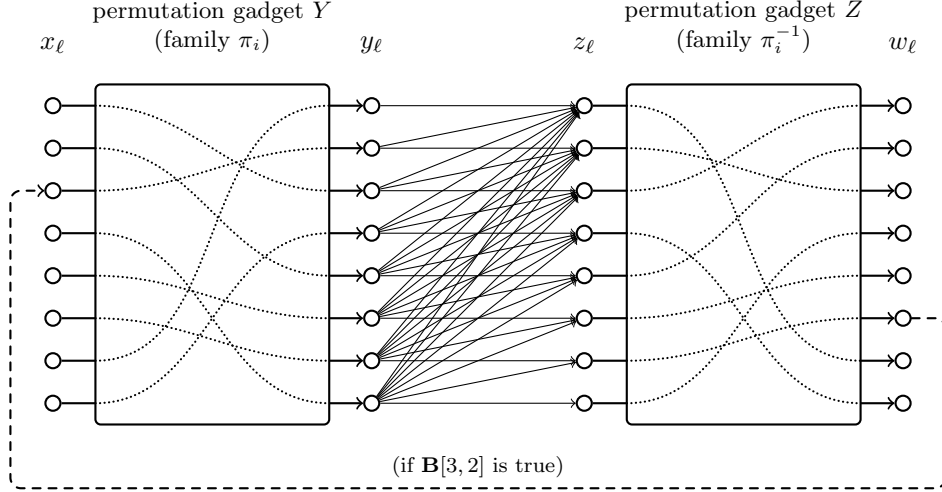


Figure 3: A depiction of the graph G_1 for $m = 4$. The dotted lines within each permutation gadget $X \in \{Y, Z\}$ represent maximal paths not passing through $K^X(\pi_i^X)$, and depend on the vertex i in the graph G_0 of Alice where her token is located. The dashed line is an example of edge (w_{m+j}, x_k) , for $j = 2$ and $k = 3$, which is present only if $\mathbf{B}[3, 2] = 1$. The actual graph has a similar edge for every k and j such that $\mathbf{B}[k, j] = 1$.

Lemma 22. *If $\mathbf{A}, \mathbf{B}, \mathbf{C}$ comprise an invalid triangle (i, j, k) , then $\langle 0, i, x_k \rangle \in \mathcal{S}^1$.*

Proof. First of all, Bob has to move his token following the permutation networks Y and Z . Namely, for $X \in \{Y, Z\}$, if his token is on a non-last vertex u in one of the maximal paths of X not passing through $K^X(\pi_i^X)$, then Bob has to move his token to the next vertex u' in the path. This rule is necessary and sufficient to ensure that the Alice cannot stop the play on \mathcal{F}^0 and win.

To close a cycle, Bob moves first his token along the permutation gadget Y , from x_k to $y_{\pi_i(k)}$. Then, he moves it from $y_{\pi_i(k)}$ to $z_{\pi_i(m+j)}$, which is possible since $\mathbf{A}[i, k] > \mathbf{C}[i, j]$, so $\pi_i(k) > \pi_i(m+j)$ and the edge $(y_{\pi_i(k)}, z_{\pi_i(m+j)})$ is given by construction. Next, he follows the permutation gadget Z , moving the token from $z_{\pi_i(m+j)}$ to $w_{\pi_i^{-1}(\pi_i(m+j))} = w_{m+j}$, and, finally, he moves from w_{m+k} back to x_j , which is possible since $\mathbf{B}[k, j] = 1$. \square

Lemma 23. *If $\mathbf{A}, \mathbf{B}, \mathbf{C}$ do not comprise an invalid triangle, then $\mathcal{W}^0 = \mathcal{V}$.*

Proof. The token of Alice starts on $i \in V_0$. If Bob moves his token to a vertex in either $K^Y(\pi_i)$ or $K^Z(\pi_i^{-1})$, then Alice stops at the next turn and wins immediately. Since Bob does not have at all the opportunity to stop and win the game (since $\mathcal{F}^1 \cap \mathcal{V}^1 = \emptyset$), we are left with the question whether there is an infinite play in which the token of Bob never passes through $K^Y(\pi_i)$ or $K^Z(\pi_i^{-1})$. We define a potential $p: V_1 \rightarrow \mathbb{N}$ over the possible locations of Bob's token, showing that it never increases along this hypothetical infinite play, and strictly decreases frequently, a contradiction.

Let $P_{i,\ell}^Y$ be the only maximal path in Y that goes from x_ℓ to $y_{\pi_i(\ell)}$ and does not contain any vertex in $K^Y(\pi_i^Y)$. Similarly, for the second permutation gadget, let $P_{i,\ell}^Z$ be the only maximal path in Z that goes from $z_{\pi_i(\ell)}$ to w_ℓ and does not contain any vertex in $K^Z(\pi_i^Z)$. For every vertex u along the paths $P_{i,\ell}^Y$ and $P_{i,\ell}^Z$ (including the endpoints $x_\ell, y_{\pi_i(\ell)}, z_{\pi_i(\ell)}$ and w_ℓ), let $p(u) = \pi_i(\ell)$. The only possible moves are either (a) along a path $P_{i,\ell}^Y$ or $P_{i,\ell}^Z$, where the potential remains constant by definition, or (b) from y_ℓ to $z_{\ell'}$, where $p(y_\ell) = \ell' \leq \ell = p(z_{\ell'})$, or (c) from w_{m+j} to x_k . In case (c), we have $p(w_{m+k}) = \pi_i(m+k)$ and $p(x_j) = \pi_i(j)$. Since there are no invalid triangles and $\mathbf{B}[k, j] = 1$,

necessarily $\mathbf{A}[i, k] \leq \mathbf{C}[i, j]$, so $\pi_i(m+k) < \pi_i(j)$, and the potential strictly decreases. Furthermore, moves of type (c) occur frequently since without them the configuration graph is acyclic. \square

References

- [1] Amir Abboud and Virginia Vassilevska Williams. Popular conjectures imply strong lower bounds for dynamic problems. In *2014 IEEE 55th Annual Symposium on Foundations of Computer Science*, pages 434–443. IEEE, October 2014.
- [2] Rajeev Alur, Thomas A. Henzinger, and Orna Kupferman. Alternating-time temporal logic. *Journal of the ACM*, 49(5):672–713, September 2002.
- [3] Christel Baier, Joost-Pieter Katoen, and Kim Guldstrand Larsen. *Principles of model checking*. 2008.
- [4] Catriel Beeri. On the membership problem for functional and multivalued dependencies in relational databases. *ACM Transactions on Database Systems*, 5(3):241–259, September 1980.
- [5] Václav Edvard Beneš. *Mathematical theory of connecting networks and telephone traffic*. 1965.
- [6] Bard Bloom. Ready simulation, bisimulation, and the semantics of ccs-like languages. 1989.
- [7] Bard Bloom and Robert Paige. Transformational design and implementation of a new efficient solution to the ready simulation problem. *Science of Computer Programming*, 24(3):189–220, June 1995.
- [8] Doron Bustan and Orna Grumberg. Simulation-based minimization. *ACM Transactions on Computational Logic*, 4(2):181–206, 2003.
- [9] Gérard Cécé. Three simulation algorithms for labelled transition systems. pages 1–26, January 2013.
- [10] Pavol Černý, Thomas A Henzinger, and Arjun Radhakrishna. Simulation distances. In Paul Gastin and François Laroussinie, editors, *CONCUR 2010 - Concurrency Theory: 21th International Conference, CONCUR 2010, Paris, France, August 31-September 3, 2010. Proceedings*, volume 6269 of *Lecture Notes in Computer Science*, pages 253–268. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.
- [11] Krishnendu Chatterjee and Monika Henzinger. Efficient and dynamic algorithms for alternating büchi games and maximal end-component decomposition. *Journal of the ACM*, 61(3):1–40, June 2014.
- [12] Rance Cleaveland, Joachim Parrow, and Bernhard Steffen. The concurrency workbench: a semantics-based tool for the verification of concurrent systems. *ACM Transactions on Programming Languages and Systems*, 15(1):36–72, January 1993.
- [13] Rance Cleaveland and Bernhard Steffen. A linear-time model-checking algorithm for the alternation-free modal mu-calculus. *Formal Methods in System Design*, 2(2):121–147, April 1993.
- [14] Charles Clos. A study of non-blocking switching networks. *Bell System Technical Journal*, 32(2):406–424, March 1953.

- [15] Don Coppersmith and Shmuel Winograd. Matrix multiplication via arithmetic progressions. In *Proceedings of the nineteenth annual ACM conference on Theory of computing - STOC '87*, pages 1–6, New York, New York, USA, 1987. ACM Press.
- [16] David Culler, Jaswinder Pal Singh, and Anoop Gupta. *Parallel Computer Architecture: A Hardware/Software Approach*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1998.
- [17] William James Dally and Brian Patrick Towles. *Principles and Practices of Interconnection Networks*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2004.
- [18] L. de Alfaro, T.A. Henzinger, and O. Kupferman. Concurrent reachability games. In *Proceedings 39th Annual Symposium on Foundations of Computer Science*, pages 564–575. IEEE Comput. Soc, 1998.
- [19] Ran Duan and Seth Pettie. Fast algorithms for (max, min)-matrix multiplication and bottleneck shortest paths. In *Proceedings of the Twentieth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '09, pages 384–391, Philadelphia, PA, USA, 2009. Society for Industrial and Applied Mathematics.
- [20] Kousha Etessami, Thomas Wilke, and Rebecca A. Schuller. Fair simulation relations, parity games, and state space reduction for büchi automata. *SIAM Journal on Computing*, 34(5):1159–1175, January 2005.
- [21] Rusins Freivalds. Probabilistic machines can use less running time. In *Information Processing 77, Proceedings of IFIP Congress 77*, pages 839–842, 1977.
- [22] Raffaella Gentilini, Carla Piazza, and Alberto Policriti. Simulation as coarsest partition problem. In *Tools and Algorithms for the Construction*, pages 415–430. 2002.
- [23] Raffaella Gentilini, Carla Piazza, and Alberto Policriti. From bisimulation to simulation: Coarsest partition problems. *Journal of Automated Reasoning*, 31(1):73–103, 2003.
- [24] Monika R. Henzinger, Thomas A. Henzinger, and Peter .W. Kopke. Computing simulations on finite and infinite graphs. In *Proceedings of IEEE 36th Annual Foundations of Computer Science*, pages 453–462. IEEE Comput. Soc. Press, 1995.
- [25] C. A. R. Hoare. Communicating sequential processes. In *The Origin of Concurrent Programming*, pages 413–443. Springer New York, New York, NY, 1978.
- [26] Marcin Jurdziński. Small progress measures for solving parity games. In Horst Reichel and Sophie Tison, editors, *17th Annual Symposium on Theoretical Aspects of Computer Science (STACS), Proceedings*, pages 290–301. Springer Berlin Heidelberg, Lille, France, 2000.
- [27] Tracy Kimbrel and Rakesh Kumar Sinha. A probabilistic algorithm for verifying matrix products using $o(n + 2)$ time and $\log 2n + o(1)$ random bits. *Information Processing Letters*, 45(2):107–110, February 1993.
- [28] Ivan Korec and Jiří Wiedermann. Deterministic verification of integer matrix multiplication in quadratic time. In *SOFSEM 2014: Theory and Practice of Computer*, pages 375–382. 2014.
- [29] Saul A. Kripke. Semantical considerations on modal logic. *Acta Philosophica Fennica*, (16):83–94, 1963.

- [30] Jasen Markovski. Saving time in a space-efficient simulation algorithm. In *2011 11th International Conference on Quality Software*, pages 244–251. IEEE, July 2011.
- [31] R. Milner. An algebraic definition of simulation between programs. In *Proceedings of the 2nd International Joint Conferences on Artificial Intelligence (IJCAI)*, pages 481–489, London, 1971.
- [32] Robert Paige and Robert E. Tarjan. Three partition refinement algorithms. *SIAM Journal on Computing*, 16(6):973–989, December 1987.
- [33] David Park. Concurrency and automata on infinite sequences. In *Theoretical Computer Science*, volume 104, pages 167–183. Springer-Verlag, Berlin/Heidelberg, 1981.
- [34] Francesco Ranzato and Francesco Tapparo. A new efficient simulation equivalence algorithm. In *22nd Annual IEEE Symposium on Logic in Computer Science (LICS 2007)*, pages 171–180. IEEE, 2007.
- [35] Francesco Ranzato and Francesco Tapparo. An efficient simulation algorithm based on abstract interpretation. *Information and Computation*, 208(1):1–22, January 2010.
- [36] L. J. Stockmeyer and A. R. Meyer. Word problems requiring exponential time (preliminary report). In *Proceedings of the fifth annual ACM symposium on Theory of computing - STOC '73*, pages 1–9, New York, New York, USA, 1973. ACM Press.
- [37] Virginia Vassilevska, Ryan Williams, and Raphael Yuster. All pairs bottleneck paths and max-min matrix products in truly subcubic time. *Theory of Computing*, 5(1):173–189, 2009.
- [38] Virginia Vassilevska W. and Ryan Williams. Subcubic equivalences between path, matrix and triangle problems. In *2010 IEEE 51st Annual Symposium on Foundations of Computer Science*, pages 645–654. IEEE, October 2010.
- [39] Abraham Waksman. A permutation network. *Journal of the ACM*, 15(1):159–163, 1968.

A Proof of properties of reachability games

Proof of Proposition 4. Each property is proven below.

- (a) Take $s(\sigma) \in \mathcal{U}$ if $\sigma \in \text{LiftSet}^P(\mathcal{U}) \cap \mathcal{V}^P \cap \mathcal{F}^{1-P}$ (by definition of LiftSet^P , a move $(\sigma, \sigma') \in \mathcal{E}$ with $\sigma' \in \mathcal{U}$ exists) and $s(\sigma) = \perp$ otherwise. Let $\pi = \sigma_0 \sigma_1 \dots$ be a play starting with $\sigma_0 \in \mathcal{U}$ and conforming to s . Then, the play π remains on \mathcal{U} , and it is either winning for P or infinite. Indeed, if $\sigma_i \in \mathcal{U} \subseteq \text{LiftSet}^P(\mathcal{U})$, then either σ_i is the final configuration in π and $\sigma_i \in \mathcal{F}^P$, or $\sigma_{i+1} \in \mathcal{U}$ for the next configuration σ_{i+1} .
- (b) For $\sigma \in \mathcal{V}^P$, let $s(\sigma) = \perp$ if $p(\sigma) = 0$, and take $s(\sigma)$ such that $p(s(\sigma)) < \text{Lift}^P(p)(\sigma) \leq p(\sigma)$ if $1 \leq p(\sigma) < \infty$ (such a $s(\sigma)$ exists by definition of Lift^P). For any play $\pi = \sigma_0 \sigma_1 \dots$ conforming to s with $p(\sigma_0) < \infty$, the value $p(\sigma_i)$ decreases strictly with i , so π is finite. Let σ_ℓ be the last configuration of π . If $\sigma_\ell \in \mathcal{V}^P$, then $p(\sigma_\ell) = 0$, otherwise $\sigma_\ell \in \mathcal{V}^{1-P}$ and $p(\sigma_\ell) < \infty$: in either case $\sigma_\ell \in \mathcal{F}^P$ follows.
- (d) Observe that $\text{Lift}^P(p)$ is defined equivalently by the equations $\{\sigma \mid \text{Lift}^P(p)(\sigma) < k + 1\} = \text{LiftSet}^P(\{\sigma \mid p(\sigma) < k\})$ for $k \in \mathbb{N}$. Hence, construct recursively the sets $\mathcal{W}_{<k+1}^P = \text{LiftSet}^P(\mathcal{W}_{<k}^P)$, for every $k \in \mathbb{N}$, starting with $\mathcal{W}_{<0}^P = \emptyset$. Clearly $\mathcal{W}_{<0}^P \subseteq \mathcal{W}_{<1}^P$, and

by induction $\mathcal{W}_{<k}^P \subseteq \mathcal{W}_{<k+1}^P$ for every $k \in \mathbb{N}$ since $LiftSet^P$ is monotone. Now define $\mathcal{W}_k^P = \mathcal{W}_{<k+1}^P \setminus \mathcal{W}_{<k}^P$, for $k \in \mathbb{N}$, observing that they are pairwise disjoint, and let $r^P(\sigma) = k$, if $\sigma \in \mathcal{W}_k^P$ for some k , or $r^P(\sigma) = \infty$ otherwise. The potential r^P is the unique solution of the equation $Lift^P(p) = p$ by construction. Indeed, $\{\sigma \mid r^P(\sigma) < k+1\} = LiftSet^P(\{\sigma \mid r^P(\sigma) < k\})$ for every $k \in \mathbb{N}$, so $Lift^P(r^P) = r^P$. Suppose to have a distinct solution $Lift^P(p) = p$ and take the smallest k such that $\{\sigma \mid p(\sigma) = k\} \neq \mathcal{W}_k^P$. Then $\{\sigma \mid p(\sigma) < k\} = \mathcal{W}_{<k}^P$ so $\{\sigma \mid p(\sigma) < k+1\} = LiftSet^P(\{\sigma \mid p(\sigma) < k\}) = LiftSet^P(\mathcal{W}_{<k}^P) = \mathcal{W}_{<k+1}^P$ and $\{\sigma \mid p(\sigma) = k\} = \{\sigma \mid p(\sigma) < k+1\} \setminus \mathcal{W}_{<k}^P = \mathcal{W}_k^P$. Since r^P is a progress measure, $\text{supp}(r^P) \subseteq \mathcal{W}^P$. Observe that $LiftSet^{1-P}(\mathcal{V} \setminus \text{supp}(r^P)) = \mathcal{V} \setminus LiftSet^P(\text{supp}(r^P)) = \mathcal{V} \setminus \text{supp}(r^P)$, so $\mathcal{V} \setminus \text{supp}(r^P)$ is closed for $1-P$ and contained in \mathcal{S}^{1-P} . As \mathcal{W}^P and \mathcal{S}^{1-P} are clearly disjoint, we get the statement.

- (c) We already showed that \mathcal{W}^P and \mathcal{S}^P are fixpoints of $LiftSet^P$. Being closed, any other fixpoint is contained in \mathcal{S}^P by (a), so \mathcal{S}^P is the greatest fixpoint. By induction $\mathcal{W}_{<k}^P$ is contained in every fixpoint, since $\mathcal{W}_{<0}^P = \emptyset$ and $\mathcal{W}_{<k+1}^P = LiftSet^P(\mathcal{W}_{<k}^P)$. Thus \mathcal{W}^P is the least fixpoint.
- (e) Let $\delta^+(\sigma)$ be the number of moves $(\sigma, \sigma') \in \mathcal{E}$, and let $c_k(\sigma) \leq \delta^+(\sigma)$ be the number of moves $(\sigma, \sigma') \in \mathcal{E}$ such that $\sigma' \in \mathcal{W}_{<k}^P$. We can characterize $\mathcal{W}_{<k+1}^P$ as follows: for $\sigma \in \mathcal{V}_P$, we have $\sigma \in \mathcal{W}_{<k+1}^P$ iff $\sigma \in \mathcal{F}^P$ or $c_k(\sigma) > 0$, while for $\sigma \in \mathcal{V}_{1-P}$, we have $\sigma \in \mathcal{W}_{<k+1}^P$ iff $\sigma \in \mathcal{F}^P$ and $c_k(\sigma) = \delta^+(\sigma)$. Maintain a counter $c: \mathcal{V} \rightarrow \mathbb{N}$. Start with $c(\sigma) = c_0(\sigma) = 0$ for every $\sigma \in \mathcal{V}$, and compute the set $\mathcal{W}_0^P = (\mathcal{V}^P \cap \mathcal{F}^P) \cup \{\sigma \in \mathcal{V}^{1-P} \cap \mathcal{F}^P \mid \delta^+(\sigma) = 0\}$ in $O(|\mathcal{V}|)$ time. Then, for each $k = 1, \dots, |\mathcal{V}| - 1$, compute c_k and \mathcal{W}_k^P as follows: for each move $(\sigma, \sigma') \in \mathcal{E}$ with $\sigma' \in \mathcal{W}_{k-1}^P$, increase the value of $c(\sigma)$ by one, so that at the end $c(\sigma) = c_k(\sigma)$ for every $\sigma \in \mathcal{V}$. If a configuration σ satisfies for the first time the condition $c_k(\sigma) > 0$ (if $\sigma \in \mathcal{V}^P$) or $c_k(\sigma) = \delta^+(\sigma)$ (if $\sigma \in \mathcal{V}^{1-P} \cap \mathcal{F}^P$), then add σ to \mathcal{W}_k^P . Since visiting a move is done in constant time, and each move is visited at most once, this phase requires $O(\mathcal{E})$ time. The total time needed is then $O(|\mathcal{V}| + |\mathcal{E}|)$.

□

B Equivalence with classical definition of simulation preorder

Definition 24 (Simulation preorder, classical definition). A binary relation $R \subseteq S \times S$ is a *simulation* if, for every $(s, t) \in R$, we have that (a) s and t have the same label $L(s) = L(t)$, and (b) for every transition $(s, s') \in T$ there is a transition $(t, t') \in T$ such that $(s', t') \in R$.

For $s, t \in S$, we say that t *simulates* s (written $s \preceq_s t$) if there exists a simulation R with $(s, t) \in R$.

Proposition 25 ([24, 20, 10]). *Definition 7 and Definition 24 of simulation preorder are equivalent.*

Proof. (\implies) Take a simulation relation $R \subseteq S \times S$ and define

$$\mathcal{U} = \{\langle 0, s, t \rangle \mid (s, t) \in R\} \cup \{\langle 1, t, s' \rangle \mid \exists s \text{ such that } (s, t) \in R \text{ and } (s, s') \in T\}.$$

We prove that \mathcal{U} is closed on $(\mathcal{G}, \mathcal{F})$, so $\mathcal{U} \subseteq \mathcal{S}^1$. Take $\langle 0, s, t \rangle \in \mathcal{U}$. Since $(s, t) \in R$, we have $L(s) = L(t)$ by definition of simulation, so $\langle 0, s, t \rangle \in \mathcal{F}^1$. Moreover, for every $(\langle 0, s, t \rangle, \langle 1, t, s' \rangle) \in \mathcal{E}$ we have $(s, s') \in T$ so $\langle 1, t, s' \rangle \in \mathcal{U}$ by construction. Now take any $\langle 1, t, s' \rangle \in \mathcal{U}$ and let $s \in S$ be such that $(s, t) \in R$ and $(s, s') \in T$. By definition of simulation, there is a $t' \in S$ such that $(t, t') \in T$ and $(s', t') \in R$. Hence, $(\langle 1, t, s' \rangle, \langle 0, s', t' \rangle) \in \mathcal{E}$ with $\langle 0, s', t' \rangle \in \mathcal{U}$.

(\Leftarrow) We prove that the relation $R = \{(s, t) \mid \langle 0, s, t \rangle \in \mathcal{S}^1\}$ is a simulation. Suppose $(s, t) \in R$ so $\langle 0, s, t \rangle \in \mathcal{S}^1$. Observe that $\langle 0, s, t \rangle \in \mathcal{F}^1$, so $L(s) = L(t)$, otherwise Alice wins by stopping on $\langle 0, s, t \rangle$. For any edge $(s, s') \in T$, we have $(\langle 0, s, t \rangle, \langle 1, t, s' \rangle) \in \mathcal{E}$ and, since $\langle 0, s, t \rangle \in \mathcal{S}^1$, also $\langle 1, t, s' \rangle \in \mathcal{S}^1$. However, since $\langle 1, t, s' \rangle \notin \mathcal{F}^1$, then there exists a $t' \in S$ such that $(\langle 1, t, s' \rangle, \langle 0, s', t' \rangle) \in \mathcal{E}$ and $\langle 0, s', t' \rangle \in \mathcal{S}^1$. In particular, $(t, t') \in T$ and $(s', t') \in R$. \square

C Equivalence of simulation games and 2TRGs

Proof of Theorem 9, continues. We need to prove that $\mathcal{S}_{\mathcal{G}, \mathcal{F}}^1 = \mathcal{S}_{\mathcal{G}', \mathcal{F}'}^1$.

(\supseteq) Define the potential p on \mathcal{G}' as follows

$$\begin{aligned} p(\langle P, u, v \rangle) &= r_{\mathcal{G}, \mathcal{F}}^0(\langle P, u, v \rangle) + 1 && \text{for } \langle P, u, v \rangle \in \mathcal{V} \\ p(\langle 1, v, u^* \rangle) &= 0 && \text{for } \langle 0, u, v \rangle \in \mathcal{F}^0. \\ p(\langle P, x, y \rangle) &= \infty && \text{in any other case} \end{aligned}$$

Observe that p is a progress measure on $(\mathcal{G}', \mathcal{F}')$ for Alice. Thus $\mathcal{W}_{\mathcal{G}, \mathcal{F}}^0 = \text{supp}(p) \cap \mathcal{V} \subseteq \text{supp}(p) \subseteq \mathcal{W}_{\mathcal{G}', \mathcal{F}'}^0$.

(\subseteq) Define the set $\mathcal{U} \subseteq \mathcal{V}_{\mathcal{G}'}$ as follows

$$\begin{aligned} \langle P, u, v \rangle &\in \mathcal{U} && \text{for } \langle P, u, v \rangle \in \mathcal{S}_{\mathcal{G}, \mathcal{F}}^1 \\ \langle 1, v, u^* \rangle &\in \mathcal{U} && \text{for } \langle 0, u, v \rangle \in \mathcal{F}^1 \\ \langle 0, u, u \rangle, \langle 0, u^*, u^* \rangle &\in \mathcal{U} && \text{for } u \in V_0 \\ \langle 1, u, u' \rangle &\in \mathcal{U} && \text{for } (u, u') \in E_0 \\ \langle P, x, y \rangle &\notin \mathcal{U} && \text{in any other case} \end{aligned}$$

Observe that \mathcal{U} is closed on $(\mathcal{G}', \mathcal{F}')$ for Bob. Thus $\mathcal{S}_{\mathcal{G}, \mathcal{F}}^1 = \mathcal{U} \cap \mathcal{V} \subseteq \mathcal{U} \subseteq \mathcal{S}_{\mathcal{G}', \mathcal{F}'}^1$. \square

D Proof of dicut decomposition of reachability games

Proof of Lemma 15, continues. Denote with LiftSet_H^P and LiftSet_T^P the operators LiftSet^P on the games $(\mathcal{G}[\mathcal{V}_H], \mathcal{F}[\mathcal{V}_H])$ and $(\mathcal{G}[\mathcal{V}_T], \mathcal{F}_*[\mathcal{V}_H])$. We first need to show that $\text{LiftSet}_H^P(\mathcal{U} \cap \mathcal{V}_H) = \text{LiftSet}^P(\mathcal{U}) \cap \mathcal{V}_H$, for any $\mathcal{U} \subseteq \mathcal{V}$, and that $\text{LiftSet}_T^P(\mathcal{U}_T) = \text{LiftSet}^P(\mathcal{U}_T \cup \mathcal{S}_H^P) \cap \mathcal{V}_T$ for any $\mathcal{U}_T \subseteq \mathcal{V}_T$.

For $\mathcal{U} \subseteq \mathcal{V}$ and $\sigma \in \mathcal{V}_H \cap \mathcal{V}^P$ we have

$$\begin{aligned} \sigma \in \text{LiftSet}_H^P(\mathcal{U} \cap \mathcal{V}_H) &\iff \sigma \in \mathcal{F}^P \vee \bigvee_{(\sigma, \sigma') \in \mathcal{E}} \sigma' \in \mathcal{U} \cap \mathcal{V}_H \\ &\iff \sigma \in \text{LiftSet}^P(\mathcal{U}) \end{aligned}$$

so $\text{LiftSet}_H^P(\mathcal{U} \cap \mathcal{V}_H) \cap \mathcal{V}^P = \text{LiftSet}^P(\mathcal{U}) \cap \mathcal{V}_H \cap \mathcal{V}^P$. We obtain $\text{LiftSet}_H^P(\mathcal{U} \cap \mathcal{V}_H) = \text{LiftSet}^P(\mathcal{U}) \cap \mathcal{V}_H$ by applying de Morgan laws.

For $\mathcal{U}_T \subseteq \mathcal{V}_T$ and $\sigma \in \mathcal{V}_T \cap \mathcal{V}^P$ we have

$$\begin{aligned} \sigma \in \text{LiftSet}_T^P(\mathcal{U}_T) &\iff \sigma \in \mathcal{F}_*^P \vee \bigvee_{(\sigma, \sigma') \in \mathcal{E}} \sigma' \in \mathcal{U}_T \\ &\iff \sigma \in \text{LiftSet}^P(\mathcal{S}_H^P) \vee \bigvee_{(\sigma, \sigma') \in \mathcal{E}} \sigma' \in \mathcal{U}_T \\ &\iff \sigma \in \mathcal{F}^P \vee \bigvee_{(\sigma, \sigma') \in \mathcal{E}} \sigma' \in \mathcal{S}_H^P \vee \bigvee_{(\sigma, \sigma') \in \mathcal{E}} \sigma' \in \mathcal{U}_T \\ &\iff \sigma \in \mathcal{F}^P \vee \bigvee_{(\sigma, \sigma') \in \mathcal{E}} \sigma' \in \mathcal{U}_T \cup \mathcal{S}_H^P \\ &\iff \sigma \in \text{LiftSet}^P(\mathcal{U}_T \cup \mathcal{S}_H^P) \end{aligned}$$

so $LiftSet_T^P(\mathcal{U}_T) \cap \mathcal{V}^P = LiftSet^P(\mathcal{U}_T \cup \mathcal{S}_H^P) \cap \mathcal{V}_T \cap \mathcal{V}^P$. We obtain $LiftSet_T^P(\mathcal{U}_T) = LiftSet^P(\mathcal{U}_T \cup \mathcal{S}_H^P) \cap \mathcal{V}_T$ by applying de Morgan laws.

1. $\mathcal{S}_H^P \cup \mathcal{S}_T^P$ is closed on $(\mathcal{G}, \mathcal{F})$:

$$\begin{aligned} \mathcal{S}_H^P \cup \mathcal{S}_T^P &= LiftSet_H^P(\mathcal{S}_H^P) \cup LiftSet_T^P(\mathcal{S}_T^P) \\ &= [LiftSet^P(\mathcal{S}_H^P) \cap \mathcal{V}_H] \cup [LiftSet^P(\mathcal{S}_T^P \cup \mathcal{S}_H^P) \cap \mathcal{V}_T] \\ &\subseteq LiftSet^P(\mathcal{S}_T^P \cup \mathcal{S}_H^P) \end{aligned}$$

$$\text{so } \mathcal{S}_H^P \cup \mathcal{S}_T^P \subseteq \mathcal{S}^P,$$

2. $\mathcal{S}^P \cap \mathcal{V}_H$ is closed on $(\mathcal{G}_H, \mathcal{F}_H)$:

$$\begin{aligned} \mathcal{S}^P \cap \mathcal{V}_H &= LiftSet^P(\mathcal{S}^P) \cap \mathcal{V}_H \\ &= LiftSet_H^P(\mathcal{S}^P \cap \mathcal{V}_H) \end{aligned}$$

$$\text{so } \mathcal{S}^P \cap \mathcal{V}_H \subseteq \mathcal{S}_H^P \text{ and, together with (1), } \mathcal{S}^P \cap \mathcal{V}_H = \mathcal{S}_H^P,$$

3. $\mathcal{S}^P \cap \mathcal{V}_T$ is closed on $(\mathcal{G}_T, \mathcal{F}_T)$:

$$\begin{aligned} \mathcal{S}^P \cap \mathcal{V}_T &= LiftSet^P(\mathcal{S}^P) \cap \mathcal{V}_T \\ &= LiftSet^P(\mathcal{S}_H^P \cup (\mathcal{S}^P \cap \mathcal{V}_T)) \cap \mathcal{V}_T \\ &= LiftSet_T^P(\mathcal{S}^P \cap \mathcal{V}_T) \end{aligned}$$

$$\text{so } \mathcal{S}^P \cap \mathcal{V}_T \subseteq \mathcal{S}_T^P \text{ and, together with (1), } \mathcal{S}^P \cap \mathcal{V}_T = \mathcal{S}_T^P.$$

□